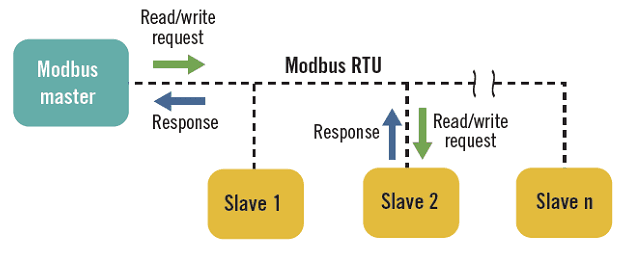
**Modbus Communication between Commercial Meter and Cortex M Microcontroller**

**1. Aim:**

To establish Modbus communication between a commercial single phase meter and Cortex M microcontroller and fetch value such as voltage and frequency.

**2. Introduction:**

Modbus is a widely adopted serial communication protocol used in industrial automation systems. It enables the exchange of data between devices such as sensors, actuators, and controllers. Modbus supports both master-slave and client-server architectures, where a master or client initiates requests, and slaves or servers respond to those requests. Modbus supports various physical communication interfaces, including RS-232, RS-485, and Ethernet (TCP/IP). In this experiment, RS-485 is utilized with the MAX 485 module to establish communication between the Cortex-M processor and the power meter.



**2.1 Signaling:**

The RS-485 interface allows for long-distance communication and multi-drop connections, making it suitable for industrial applications. It uses a differential signal to transmit data, which ensures noise immunity and reliable communication in harsh environments.

A red and blue hexagons

Description automatically generated with medium confidence

**3. Modbus Protocol:**

In Modbus communication, devices are assigned unique addresses to differentiate them. The master or client device initiates requests by sending Modbus frames to the addressed slaves or servers. The frame consists of several fields, including the slave address, function code, data, and error checking.

**3.1 Function Codes in Modbus Communication:**

Function codes define the type of operation to be performed in Modbus communication. Some commonly used function codes include:

**Read Holding Registers (Function Code: 0x03):**

This function code is used to retrieve data from holding registers. Holding registers are typically used to store measurement values, configuration parameters, or control settings. The Cortex-M processor can send a request with this function code to the power meter to obtain specific data.

**Read Input Registers (Function Code: 0x04):**

The Read Input Registers function code allows the Cortex-M processor to access data from input registers. Input registers are used to store data received from external devices or sensors. The power meter can make specific measurements or status information available through these registers.

**Write Single Register (Function Code: 0x06):**

With the Write Single Register function code, the Cortex-M processor can write a single value to a specific register in the power meter. This function is often used to update control parameters or setpoints.

**Write Multiple Registers (Function Code: 0x10):**

The Write Multiple Registers function code enables the Cortex-M processor to write multiple values to consecutive registers. This function is commonly used to update multiple parameters or send a series of commands to the power meter.

**3.2 Steps Followed in Modbus Communication:**

To retrieve data from input and holding registers using Modbus communication, the following steps are typically followed:

1. **Establish Modbus Communication**

The Cortex-M processor initializes the communication by configuring the serial port and establishing a connection with the power meter. The MAX 485 module is used to interface the processor with the RS-485 network.

1. **Frame the Modbus Request**

The Cortex-M processor constructs a Modbus request packet, specifying the function code and the starting register address. Additional parameters, such as the number of registers to read or write, can also be included in the request.

1. **Send Modbus Request:**

The Cortex-M processor transmits the Modbus request packet to the power meter via the RS-485 network. The request is sent to the address of the specific slave device.

1. **Receive Modbus Response:**

The power meter receives the Modbus request, processes it, and generates a response packet. The response packet contains the requested data from the input or holding registers. It is transmitted back to the Cortex-M processor over the RS-485 network.

1. **Extract Data from Modbus Response:**

The Cortex-M processor receives the Modbus response packet and extracts the data from the appropriate register or registers. The extracted data can then be processed, displayed, or further used for analysis or control purposes.

**3.3 Modbus Message Format:**

The Modbus protocol defines a specific message format for both requests and responses. The structure of a Modbus message consists of several fields, each serving a specific purpose. Let's explore the structure of Modbus messages for both requests and responses in the context of the above experiment.

**3.3.1 Modbus Request Message Format:**

A Modbus request message consists of the following fields:

**Slave Address:**

The Slave Address field indicates the address of the slave device (in this case, the power meter) that the request is intended for. It is a single byte value ranging from 1 to 247.

**Function Code:**

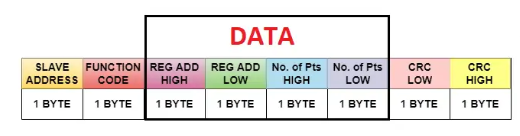
The Function Code field specifies the operation to be performed by the slave device. Different function codes represent various actions such as reading or writing data. For example, Function Code 0x03 is used for reading holding registers.

**Data:**

The Data field contains additional parameters required for the specific function code. It may include the starting register address, the number of registers to read or write, and any other necessary information.

**Error Checking:**

The Error Checking field is used to ensure the integrity of the message. It typically consists of a CRC (Cyclic Redundancy Check) or LRC (Longitudinal Redundancy Check) value calculated over the preceding fields.



**Example Modbus Request Message (Read Holding Registers)**

Let's assume we want to read two holding registers starting from address 0x100 in the power meter.

Slave Address: 0x01

Function Code: 0x03

Data: Starting Register Address: 0x0100, Number of Registers: 0x0002

Error Checking: CRC or LRC value

The Modbus request message would look like this:

| Slave Address | Function Code | Starting Register Address | Number of Registers | Error Checking |

| 0x01 | 0x03 | 0x01 |0x00| 0x00| 0x02 | CRC or LRC |

**3.3.2 Modbus Response Message Format:**

A Modbus response message also consists of specific fields:

**Slave Address:**

The Slave Address field mirrors the address specified in the corresponding request message, indicating the source of the response.

**Function Code:**

The Function Code field matches the function code of the corresponding request message, indicating the operation performed by the slave device.

**Number of Bytes:**

This contains number of bytes.

**Data:**

The Data field contains the actual data requested or processed by the slave device. In the case of a read operation, it will include the values from the requested registers.

**Error Checking:**

Similar to the request message, the Error Checking field ensures the integrity of the response message by employing CRC or LRC values.

**Example Modbus Response Message (Read Holding Registers):**

Following the previous example, let's assume the power meter responds with the values 0x1234 and 0xABCD for the two holding registers.

Slave Address: 0x01

Function Code: 0x03

Data: Register 1: 0x1234, Register 2: 0xABCD

Error Checking: CRC or LRC value

The Modbus response message would look like this:

| Slave Address | Function Code |number of bytes | Data (Register 1) | Data (Register 2) | Error Checking |

| 0x01 | 0x03 |0x02 | 0x12 | 0x34 | CRC or LRC |

| 0x01 | 0x03 | 0x02| 0xAB | 0xCD | CRC or LRC |

In this example, the power meter responds with the requested data from the holding registers, which is 0x1234 and 0xABCD. The response message reflects the slave address, function code, data values, and error checking.

**4. Circuit Connection:**

**4.1 Power Connections:**

* Connect the VCC pin of the MAX 485 module to the 5V power supply on the STM32F407 Discovery board.
* Connect the GND pin of the MAX 485 module to the ground (GND) of the board.

**4.2 UART Connections:**

* Connect the TX (Transmit) pin of the STM32F407 microcontroller (PD5) to the RO (Receiver Output) pin of the MAX 485 module.
* Connect the RX (Receive) pin of the STM32F407 microcontroller (PD6) to the DI (Driver Input) pin of the MAX 485 module.
* Connect the DE (Driver Enable) pin of the MAX 485 module to any available GPIO pin on the STM32F407 microcontroller (GPIO Output Pin PD12).
* Connect the RE (Receiver Enable) pin of the MAX 485 module to any available GPIO pin on the STM32F407 microcontroller (GPIO Output Pin PD312).

**4.3 RS-485 Connections:**

* Connect the A and B differential signal lines of the MAX 485 module to the corresponding A and B terminals of the RS-485 network.
* Connect the A and B terminals of the RS-485 network to the corresponding terminals of the SELEC single-phase power meter.

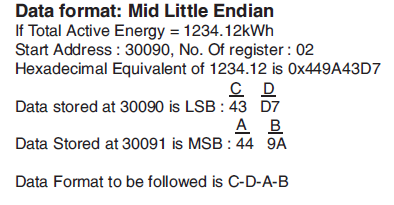
**5. Working Principle:**

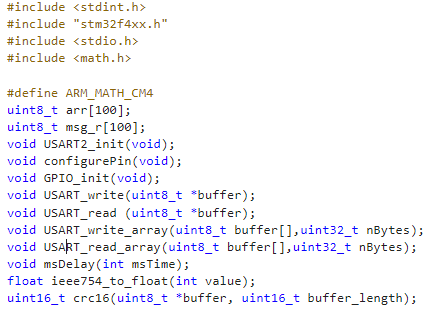
Before transmitting the request, set the DE (Driver Enable) pin of the MAX 485 module to logic HIGH, enabling the driver. Set the RE (Receiver Enable) pin of the MAX 485 module to logic LOW, disabling the receiver. Transmit the Modbus request message from the STM32F407 microcontroller to the MAX 485 module through the UART interface.

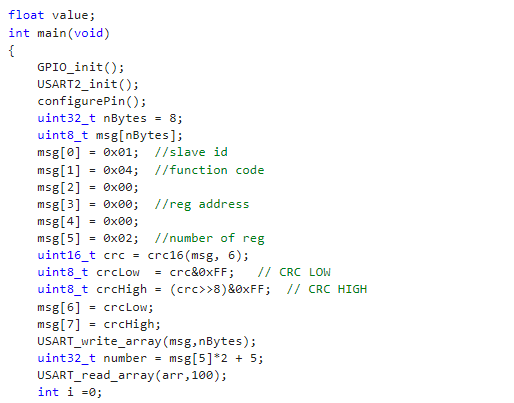
After transmitting the request, wait for the response from the power meter. Set the DE (Driver Enable) pin of the MAX 485 module to logic LOW, disabling the driver. Set the RE (Receiver Enable) pin of the MAX 485 module to logic HIGH, enabling the receiver. Receive the Modbus response message on the STM32F407 microcontroller through the UART interface.

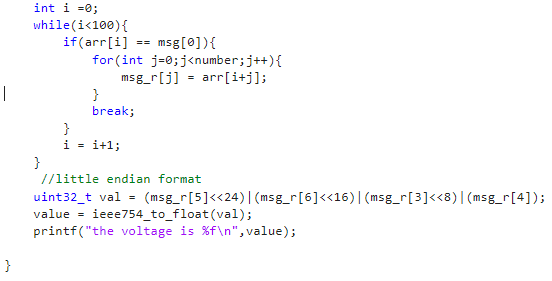
**6. Program:**

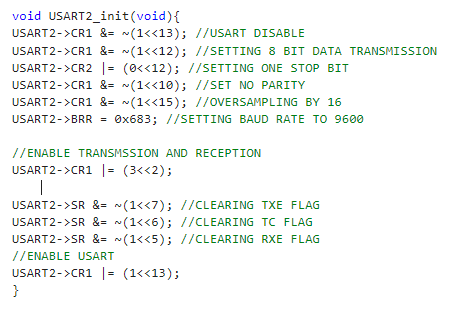
In this program we are fetching voltage value from the meter which is located at 0x0 location of input register area. It is located in 2 consecutive registers in mid little-endian format in float value (IEEE 754 format).

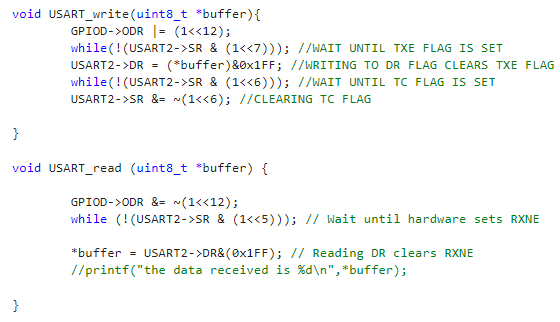


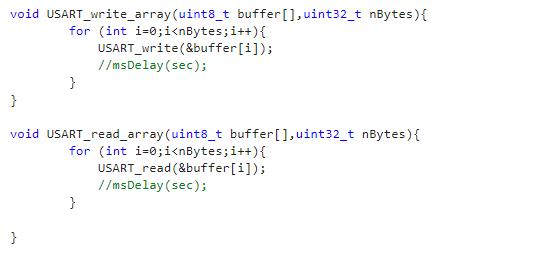












**7. Result:**

In the expression we get the value in the voltage:

